ISIS - Feature #510

Would be good to know the source for kernel data loaded by spiceinit

2011-09-30 02:23 PM - Laszlo Kestay

Status:	Closed		
Priority:	Normal		
Assignee:	Debbie Cook		
Category:	API		
Target version:	N/A		
Impact:		**Software Version:**	

Description

When you run spiceinit, especially with manually "override" kernels, it is not clear if the kernel you want is the source of the data attached to the cube. Sometimes there is another kernel that comes after and overrides the kernel you are trying to feed the cube. So a way to see which file was the source of the values attached to the cube would be helpful.

History

#1 - 2011-11-04 11:26 AM - Stuart Sides

Feasibility information has been requested from Debbie

#2 - 2011-11-05 01:21 PM - Debbie Cook

This is something I would like as well. Unfortunately from what I know of the Naif software and kernel files, the information is not available directly. According to the Naif documentation, once the kernels are loaded, they are treated as a single kernel file. I don't believe the software has any traceability from a particular segment back to its original kernel source. Just to be sure, I read through the documentation again and sent an email to the Naif guys. I will let you know what they say.

What I have done in the past for confirming a ck with a simple framing camera case is to try entering the kernels one at a time. If spiceinit runs successfully I can dump out the values and compare them to see which values match those I got from using the list. This only works for the simple cases, where only a single kernel is needed. Some of the more recent missions need multiple kernels of each type in order to complete the connections at a single instance in time. Line scan cameras have the potential to require multiple kernels to cover the time span of the cube.

Please let me know more about your particular case. Perhaps we could create a brute force utility program that would attempt to identify a ck or spk source for an instance in time. User inputs would be a cube file, kernel type (ck or spk), and a line number. It would have to cycle through the kernel list in reverse order and attempt to get data for the requested time. Once it was successful it would stop and return the file name.

One thing you can do to make sure your preferred kernel is being used whenever possible, is to make sure it is last in the list. The kernels are searched in the reverse order of which they were loaded.

I well add another note when I hear back from Naif.

#3 - 2011-11-07 11:02 AM - Debbie Cook

Below is the reply I received from Naif.

Debbie --

Unfortunately in the current SPICE system there is no way trace a

computation to specific binary and text kernels from data from which was read to do the computation. So, the best you can do right now is to give the user the list of kernels that were furnished in the order in which they were furnished either by simply saving the contents of the meta-kernels that your application loaded or by fetching the names of all loaded files using KTOTAL/KDATA.

You are not the first one who asked this question. Prompted by earlier requests like yours we discussed a number of times whether we could extend the system to collect bookkeeping information needed for such traceability but as I recall we could not find a solution that would make it possible. I'll bring this up with the team again at the next team meeting but I doubt that we will have a sudden Eureka! moment and will add this capability to the system in the near future.

Regards,
Boris.

#4 - 2011-11-09 05:04 PM - Laszlo Kestay

Personally, I would be happy to have something clear in the header that said what order spiceinit used the kernels. I know it is supposed to be in the order I specify, but there have been times when I have worried (and not known how to test) that yet another kernel was applied after my last request. To clarify, I'm not asking for the header to tell me what I input. Rather have the software add something to the header each time it calls a kernel so I know what it actually did. If that makes sense...

#5 - 2011-11-15 08:58 PM - Debbie Cook

I will try to explain the kernel search procedure and hopefully that will help. Kernels are loaded in the order they are listed in the kernels group of the label into a single virtual kernel. Once the kernel records are loaded, we have no traceability back to their original source. When a search occurs the virtual kernel is searched from the bottom up, that is, the last loaded record will be searched first. The search stops when the first successful match is found. If your targeted kernel is placed last in the list for a particular kernel type, and the desired data is in that kernel, the desired data will be read from the targeted kernel.

For most applications, the only time a kernel is "called" is when it is loaded in spiceinit. spiceinit then writes the information to the cube labels and the kernels are no longer needed.

#6 - 2012-05-21 02:34 PM - Tammy Becker

Laz, Did Debbie's last explanation help?

#7 - 2012-05-21 02:51 PM - Laszlo Kestay

Yes, the explanation explained that no change is going to be made. This was a nice-to-have but leads to the question of how one sees the kernel information (rotation matrix from the frames kernel in my case) in the label. catlab does not show me this kind of information.

#8 - 2012-10-19 06:08 PM - Debbie Cook

Laz,
I apologize for the late reply. I was not monitoring this issue and I did not know you were waiting for a reply until I went in and cleaned up my Mantis issues at Tammy's request.

You can use tabledump to see the kernel information stored in the binary tables in an Isis cube label.

To see the rotation matrix for the frames kernel, all you need to do is more or less the cube file. In the InstrumentPointing table header you will see a ConstantRotation keyword. This is typically the rotation matrix generated from the frames kernel. I pasted a sample from a MOC cube. The matrix shown rotates from Naif frame -94000 (MGS instrument platform) to -94032 (Moc WA RED)

Object = Table
Name = InstrumentPointing
StartByte = 2154181
Bytes = 5056
Records = 79
ByteOrder = Lsb
TimeDependentFrames = (-94000, 1)
ConstantFrames = (-94032, -94000)
ConstantRotation = (0.99987464549874, -0.013867590674599,
0.0076408911550023, 0.013725141347892,
0.99973667434259, 0.01839028192833,
-0.0078939080144882, -0.018283104312584,
0.99980168749255)
CkTableStartTime = -69382819.360519
CkTableEndTime = -69382512.160519

#9 - 2012-12-04 03:45 PM - Laszlo Kestay

I think this is enough information for me to figure out how to figure this out. Thanks! This ticket can be closed.

#11 - 2013-08-15 01:22 PM - Anonymous

- Target version changed from 150 to N/A